สยามชำนาญกิจ
SIAM CHAMNAN KIT

Week 02

Topics

Review RESTful API design
Develop with Spring Boot
Logging with Spring Boot
Working with Database

Q/A

```
Description:
 Web server failed to start. Port 8080 was already in use.
 Action:
 Identify and stop the process that's listening on port 8080 or configure this application to listen on another
port.
```

Stop all instances or restart IDE

server.port=<port> in file application.properties

Find and kill process

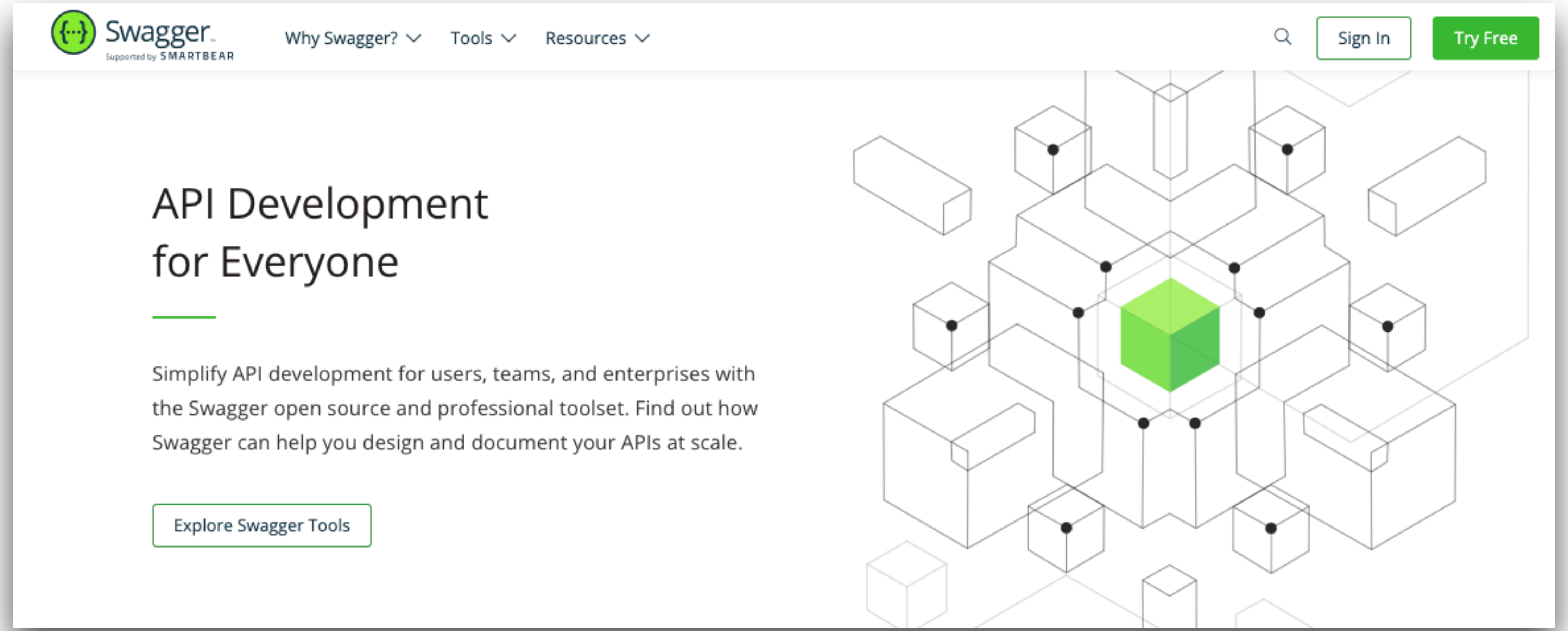
$ps -ef | grep 8080

$kill -9 <process id>

FKILL

https://www.npmjs.com/package/fkill-cli

Add swagger to Spring Boot

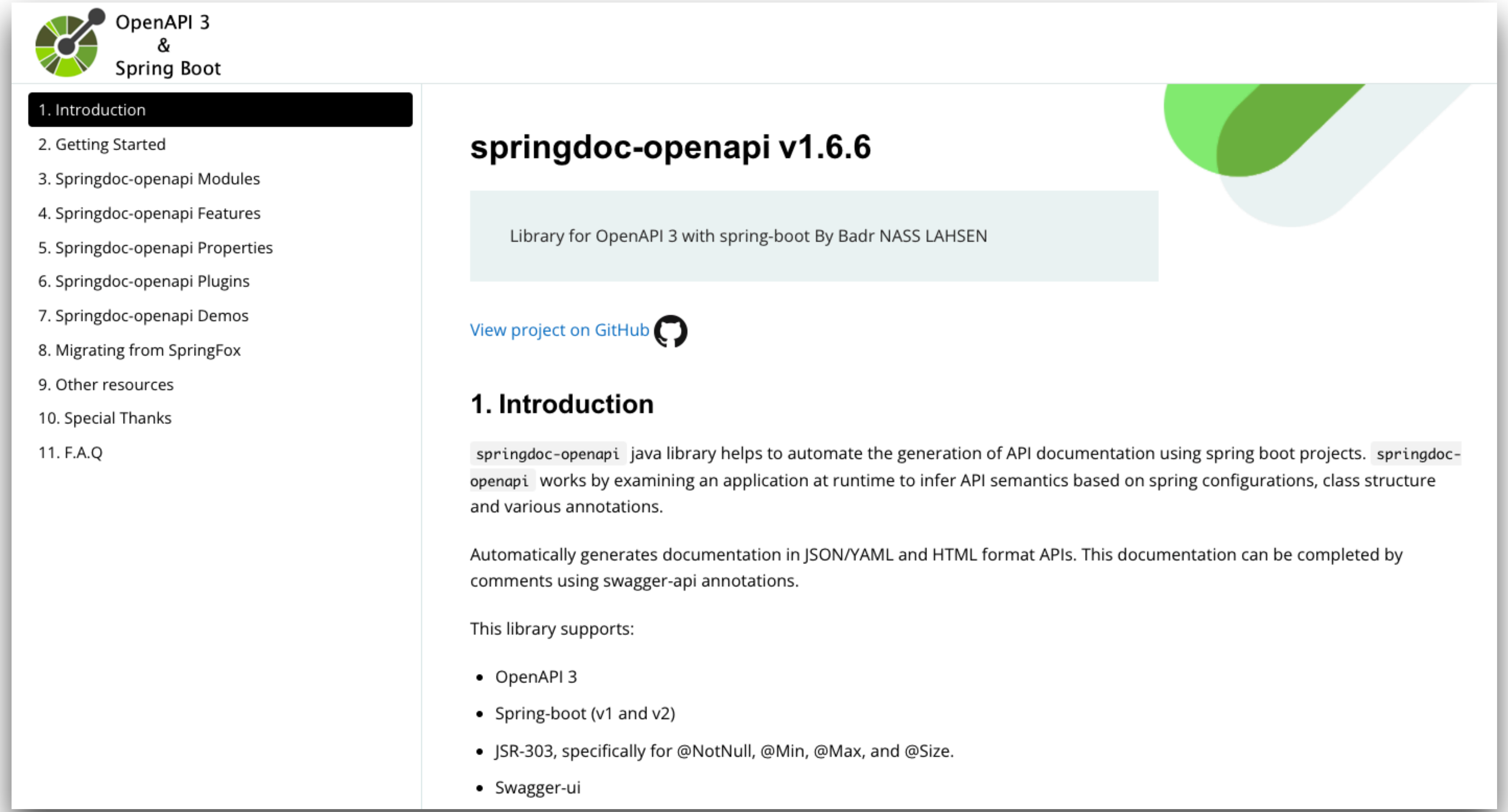


API Development for Everyone

Simplify API development for users, teams, and enterprises with the Swagger open source and professional toolset. Find out how Swagger can help you design and document your APIs at scale.

https://swagger.io/

Add springdoc-openapi



OpenAPI 3
&
Spring Boot

springdoc-openapi v1.6.6

Library for OpenAPI 3 with spring-boot By Badr NASS LAHSEN

View project on GitHub

1. Introduction

`springdoc-openapi` java library helps to automate the generation of API documentation using spring boot projects. `springdoc-openapi` works by examining an application at runtime to infer API semantics based on spring configurations, class structure and various annotations.

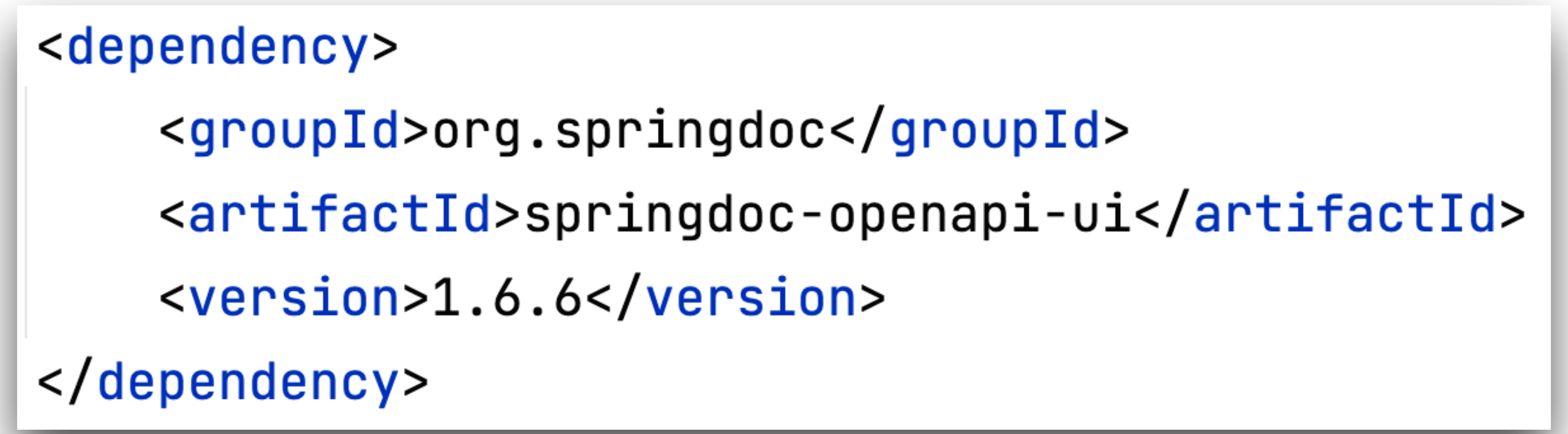
Automatically generates documentation in JSON/YAML and HTML format APIs. This documentation can be completed by comments using swagger-api annotations.

This library supports:

- OpenAPI 3
- Spring-boot (v1 and v2)
- JSR-303, specifically for @NotNull, @Min, @Max, and @Size.
- Swagger-ui

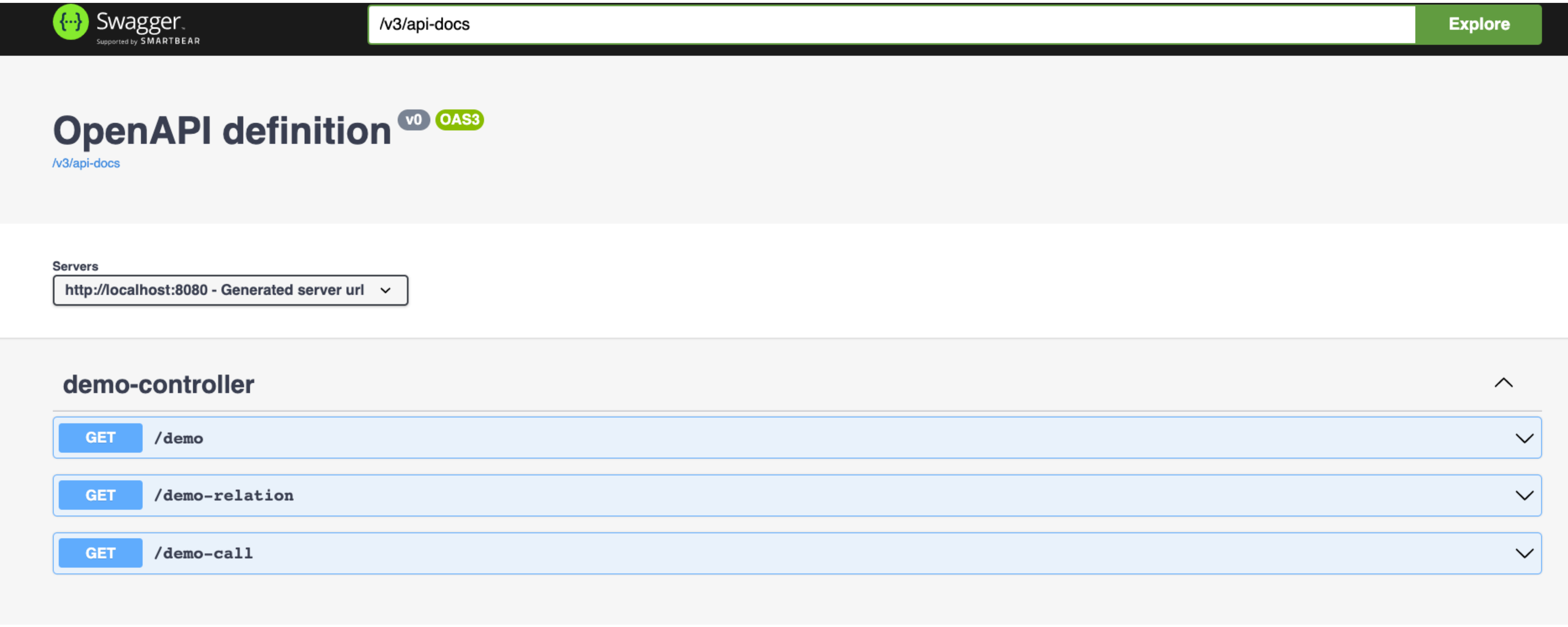
https://springdoc.org/

Add dependency



```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.6</version>
</dependency>
```

http://localhost:8080/swagger-ui.html



/v3/api-docs

OpenAPI definition v0 OAS3

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

demo-controller

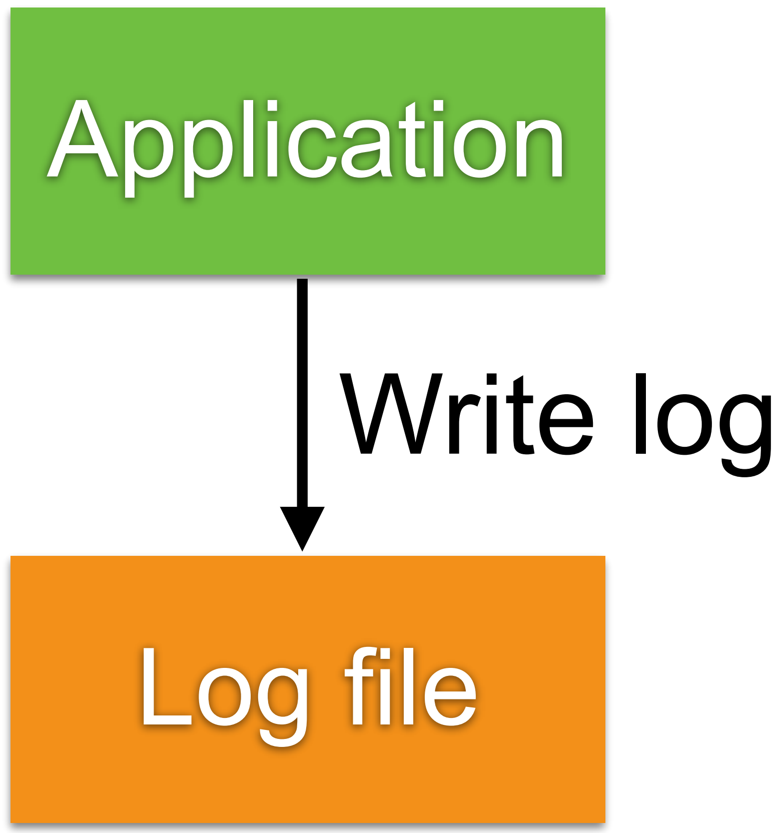
GET /demo

GET /demo-relation

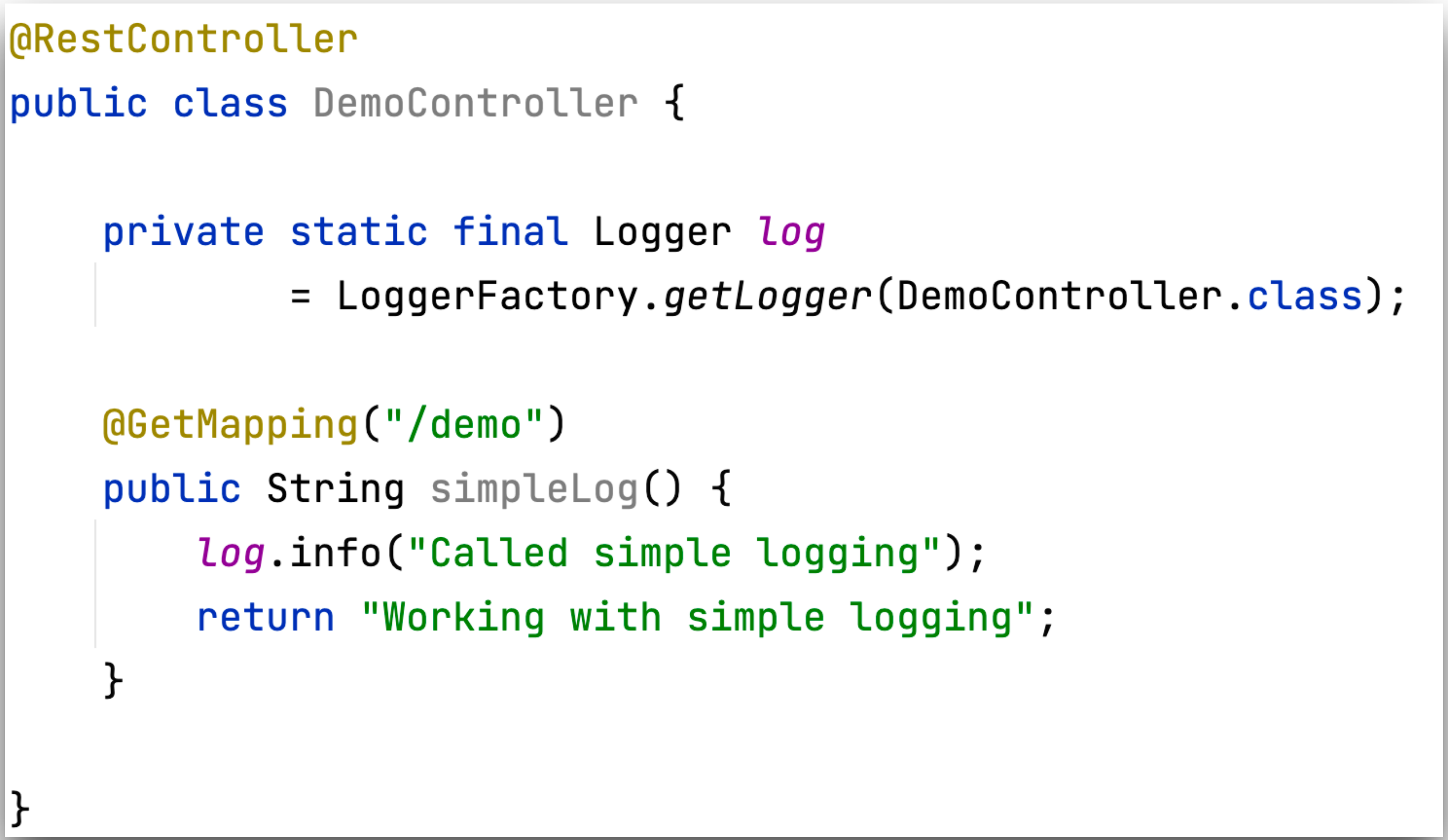
GET /demo-call

Logging with Spring Boot

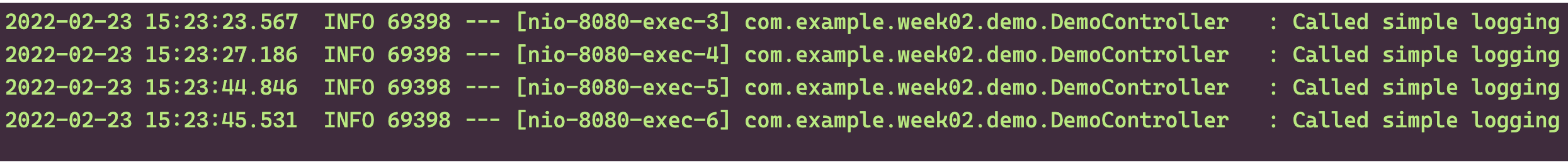
Logging with Spring boot



Logging

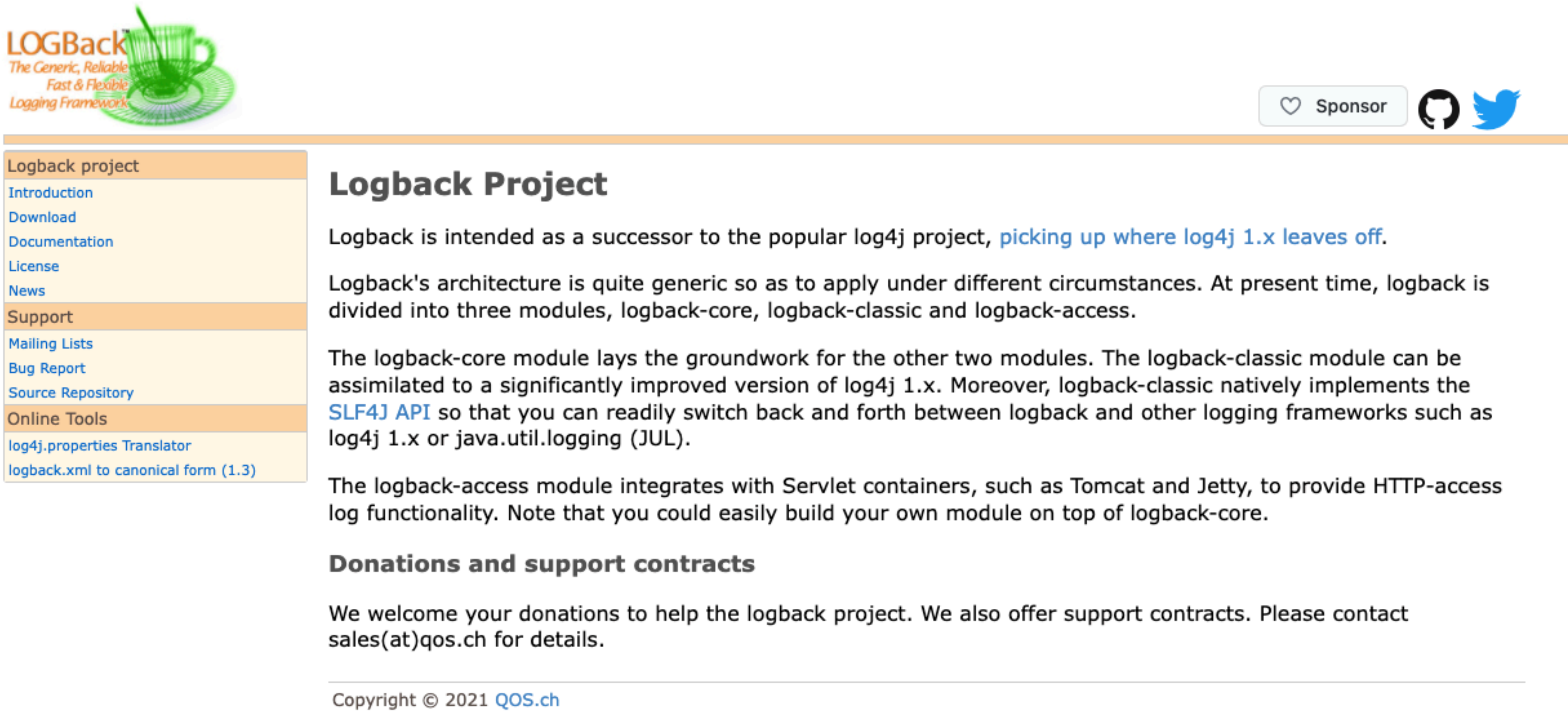


```
@RestController
public class DemoController {

    private static final Logger log
            = LoggerFactory.getLogger(DemoController.class);

    @GetMapping("/demo")
    public String simpleLog() {
        log.info("Called simple logging");
        return "Working with simple logging";
    }

}
```



```
2022-02-23 15:23:23.567  INFO 69398 --- [nio-8080-exec-3] com.example.week02.demo.DemoController   : Called simple logging
2022-02-23 15:23:27.186  INFO 69398 --- [nio-8080-exec-4] com.example.week02.demo.DemoController   : Called simple logging
2022-02-23 15:23:44.846  INFO 69398 --- [nio-8080-exec-5] com.example.week02.demo.DemoController   : Called simple logging
2022-02-23 15:23:45.531  INFO 69398 --- [nio-8080-exec-6] com.example.week02.demo.DemoController   : Called simple logging
```

Working with Logback



Logback project
- Introduction
- Download
- Documentation
- License
- News

Support
- Mailing Lists
- Bug Report
- Source Repository

Online Tools
- log4j.properties Translator
- logback.xml to canonical form (1.3)

Logback Project

Logback is intended as a successor to the popular log4j project, picking up where log4j 1.x leaves off.

Logback's architecture is quite generic so as to apply under different circumstances. At present time, logback is divided into three modules, logback-core, logback-classic and logback-access.

The logback-core module lays the groundwork for the other two modules. The logback-classic module can be assimilated to a significantly improved version of log4j 1.x. Moreover, logback-classic natively implements the SLF4J API so that you can readily switch back and forth between logback and other logging frameworks such as log4j 1.x or java.util.logging (JUL).

The logback-access module integrates with Servlet containers, such as Tomcat and Jetty, to provide HTTP-access log functionality. Note that you could easily build your own module on top of logback-core.

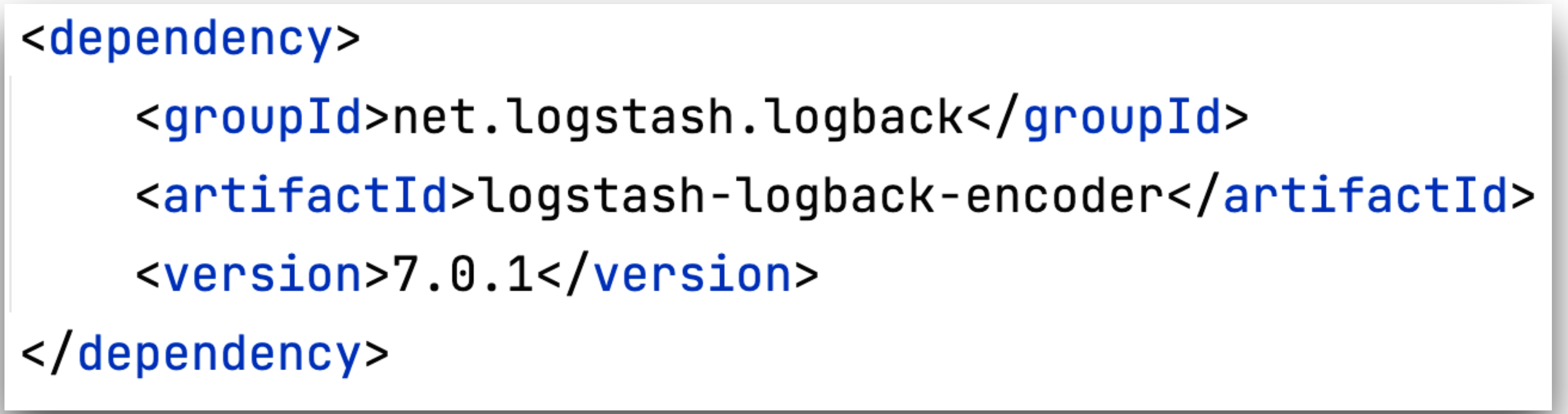
Donations and support contracts

We welcome your donations to help the logback project. We also offer support contracts. Please contact sales(at)qos.ch for details.

Copyright © 2021 QOS.ch

https://logback.qos.ch/

Add dependency



```xml
<dependency>
    <groupId>net.logstash.logback</groupId>
    <artifactId>logstash-logback-encoder</artifactId>
    <version>7.0.1</version>
</dependency>
```

Custom logging with logback

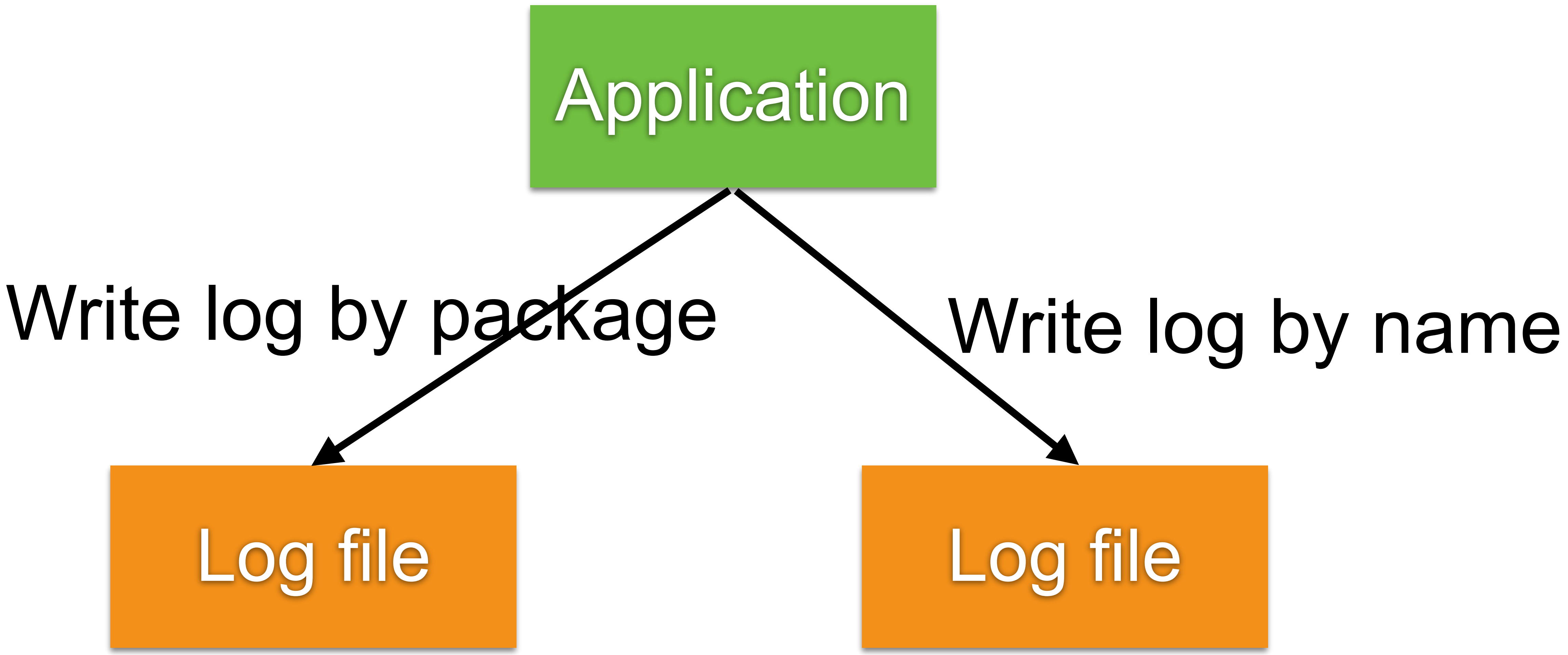
logback-spring.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>
              %d{dd-MM-yyyy HH:mm:ss.SSS} %magenta([%thread]) %highlight(%-5level) %logger{36}.%M - %msg%n
          </pattern>
        </encoder>
    </appender>
    <appender name="SAVE-TO-FILE" class="ch.qos.logback.core.FileAppender">
        <file>logs/application.log</file>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <Pattern>%d{dd-MM-yyyy HH:mm:ss.SSS} [%thread] %-5level %logger{36}.%M - %msg%n</Pattern>
        </encoder>
    </appender>
    <appender name="OUTBOUND_LOGS" class="ch.qos.logback.core.FileAppender">
        <file>logs/application-outbound.log</file>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <Pattern>%d{dd-MM-yyyy HH:mm:ss.SSS} [%thread] %-5level %logger{36}.%M - %msg%n</Pattern>
        </encoder>
    </appender>
</configuration>
```

Custom logging with logback

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

    <logger name="com.example.week02.demo" additivity="false" level="info">
        <appender-ref ref="SAVE-TO-FILE" />
        <appender-ref ref="STDOUT" />
    </logger>

    <logger name="outbound-logs" additivity="false" level="info">
        <appender-ref ref="OUTBOUND_LOGS" />
        <appender-ref ref="STDOUT" />
    </logger>

    <root level="INFO">
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

Separate log files

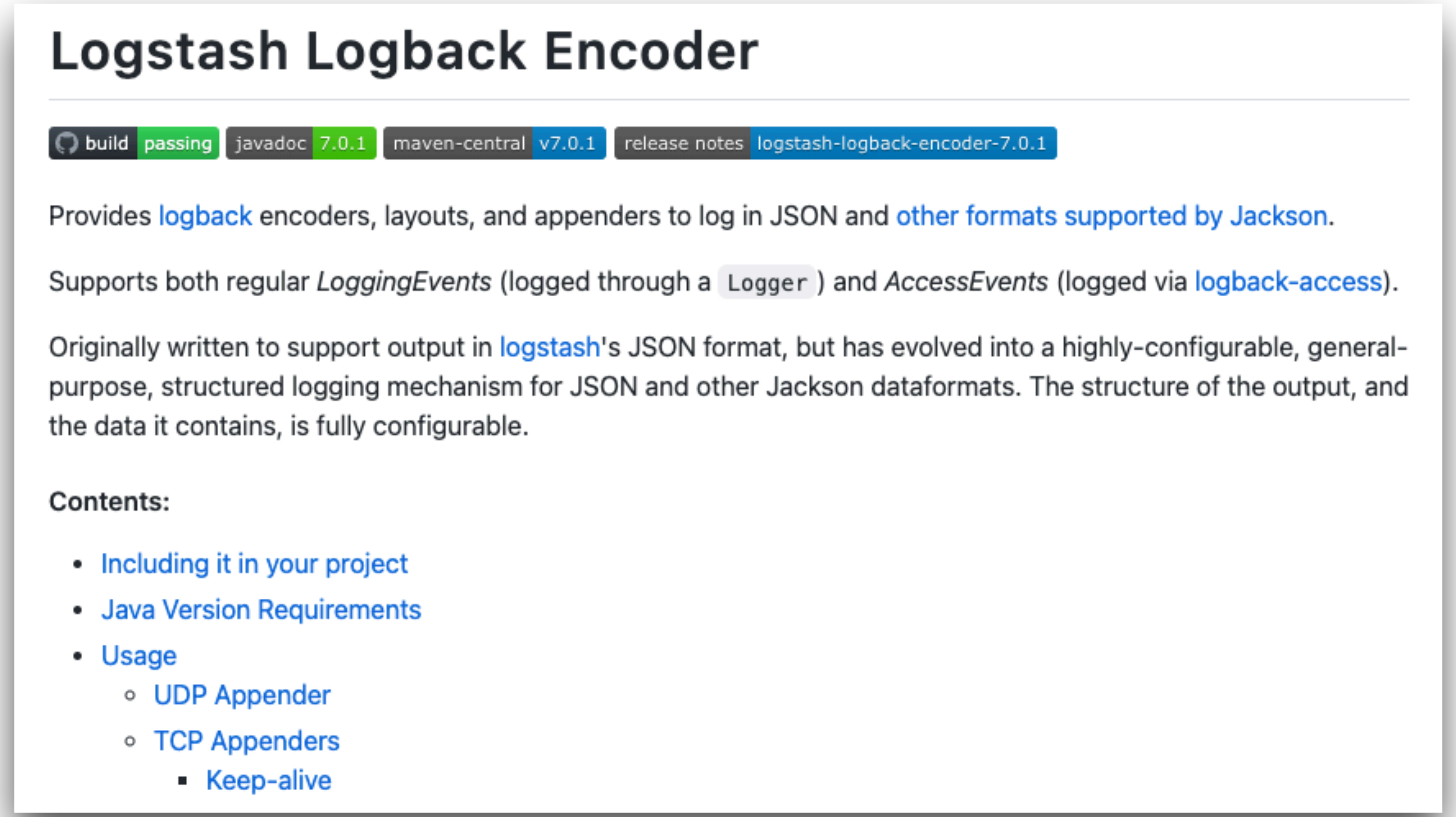


Format of log file

Default (patterns)
JSON

JSON format

Logstash Logback Encoder



Logstash Logback Encoder

build passing | javadoc 7.0.1 | maven-central v7.0.1 | release notes logstash-logback-encoder-7.0.1

Provides logback encoders, layouts, and appenders to log in JSON and other formats supported by Jackson.

Supports both regular *LoggingEvents* (logged through a `Logger`) and *AccessEvents* (logged via logback-access).

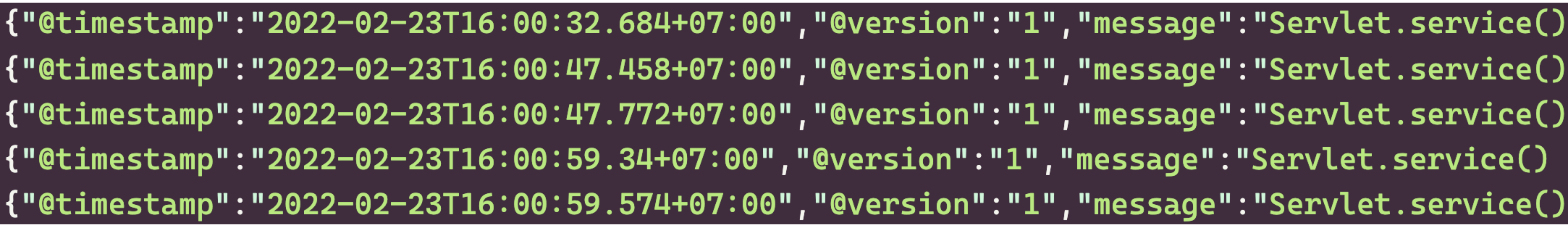
Originally written to support output in logstash's JSON format, but has evolved into a highly-configurable, general-purpose, structured logging mechanism for JSON and other Jackson dataformats. The structure of the output, and the data it contains, is fully configurable.

Contents:

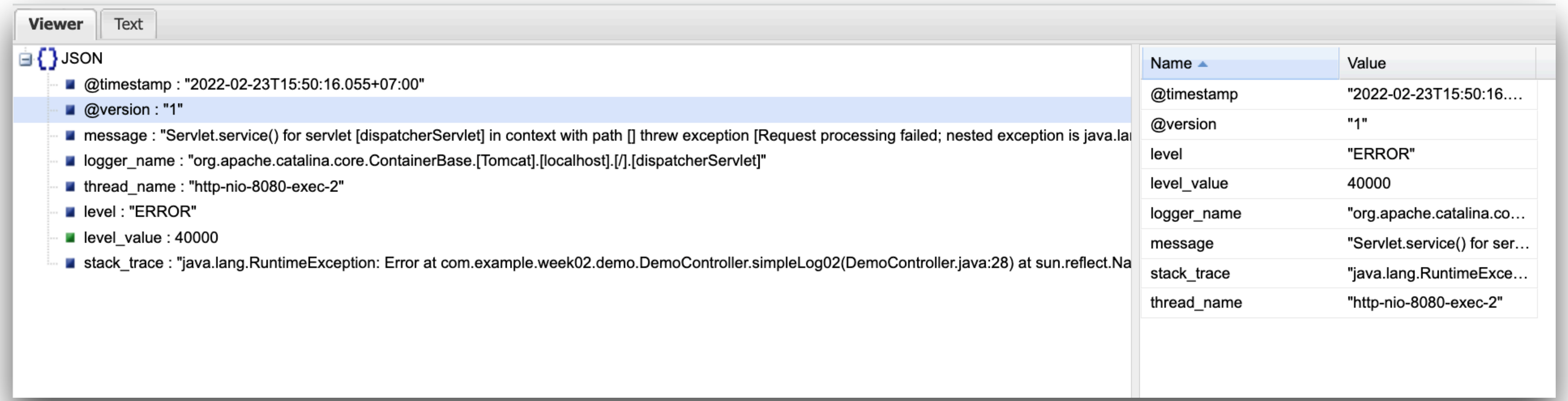
https://github.com/logfellow/logstash-logback-encoder

JSON format

Logstash Logback Encoder



```
{"@timestamp":"2022-02-23T16:00:32.684+07:00","@version":"1","message":"Servlet.service()
{"@timestamp":"2022-02-23T16:00:47.458+07:00","@version":"1","message":"Servlet.service()
{"@timestamp":"2022-02-23T16:00:47.772+07:00","@version":"1","message":"Servlet.service()
{"@timestamp":"2022-02-23T16:00:59.34+07:00","@version":"1","message":"Servlet.service()
{"@timestamp":"2022-02-23T16:00:59.574+07:00","@version":"1","message":"Servlet.service()
```

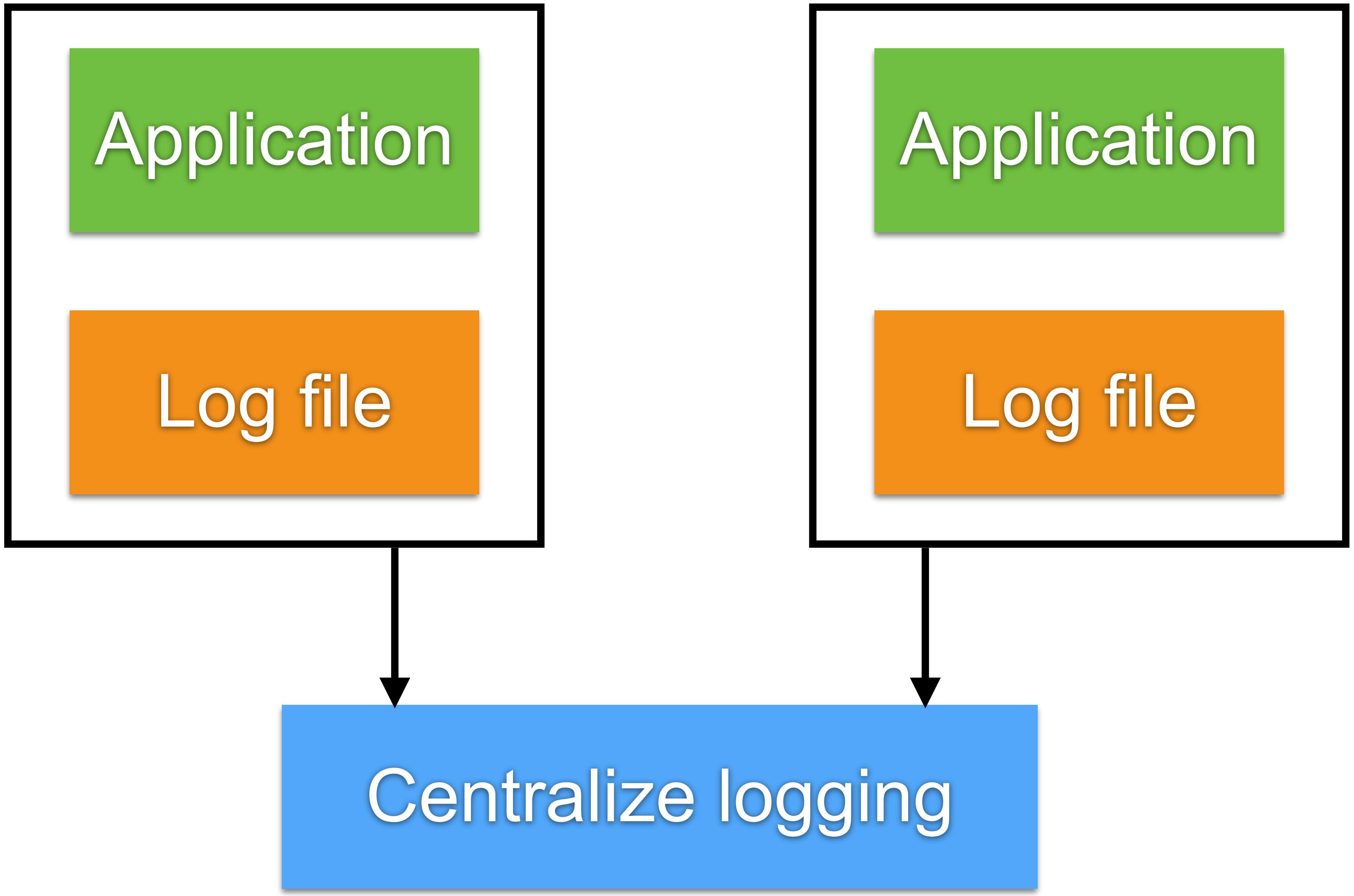


```
{} JSON
   @timestamp : "2022-02-23T15:50:16.055+07:00"
   @version : "1"
   message : "Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed; nested exception is java.la
   logger_name : "org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/].[dispatcherServlet]"
   thread_name : "http-nio-8080-exec-2"
   level : "ERROR"
   level_value : 40000
   stack_trace : "java.lang.RuntimeException: Error at com.example.week02.demo.DemoController.simpleLog02(DemoController.java:28) at sun.reflect.Na
```

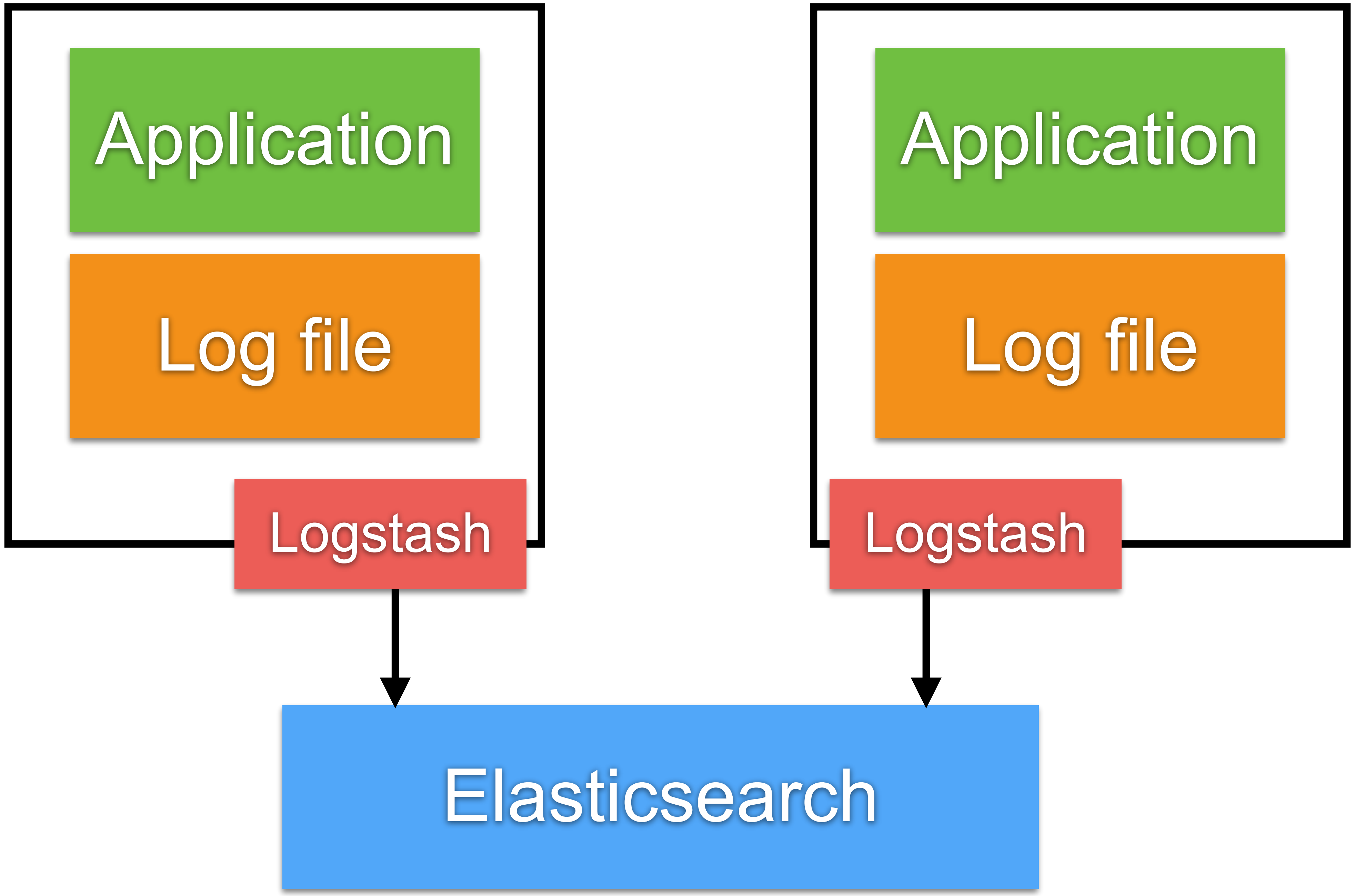
Name ▲	Value
@timestamp	"2022-02-23T15:50:16.…
@version	"1"
level	"ERROR"
level_value	40000
logger_name	"org.apache.catalina.co…
message	"Servlet.service() for ser…
stack_trace	"java.lang.RuntimeExce…
thread_name	"http-nio-8080-exec-2"

http://jsonviewer.stack.hu/

Centralize logging

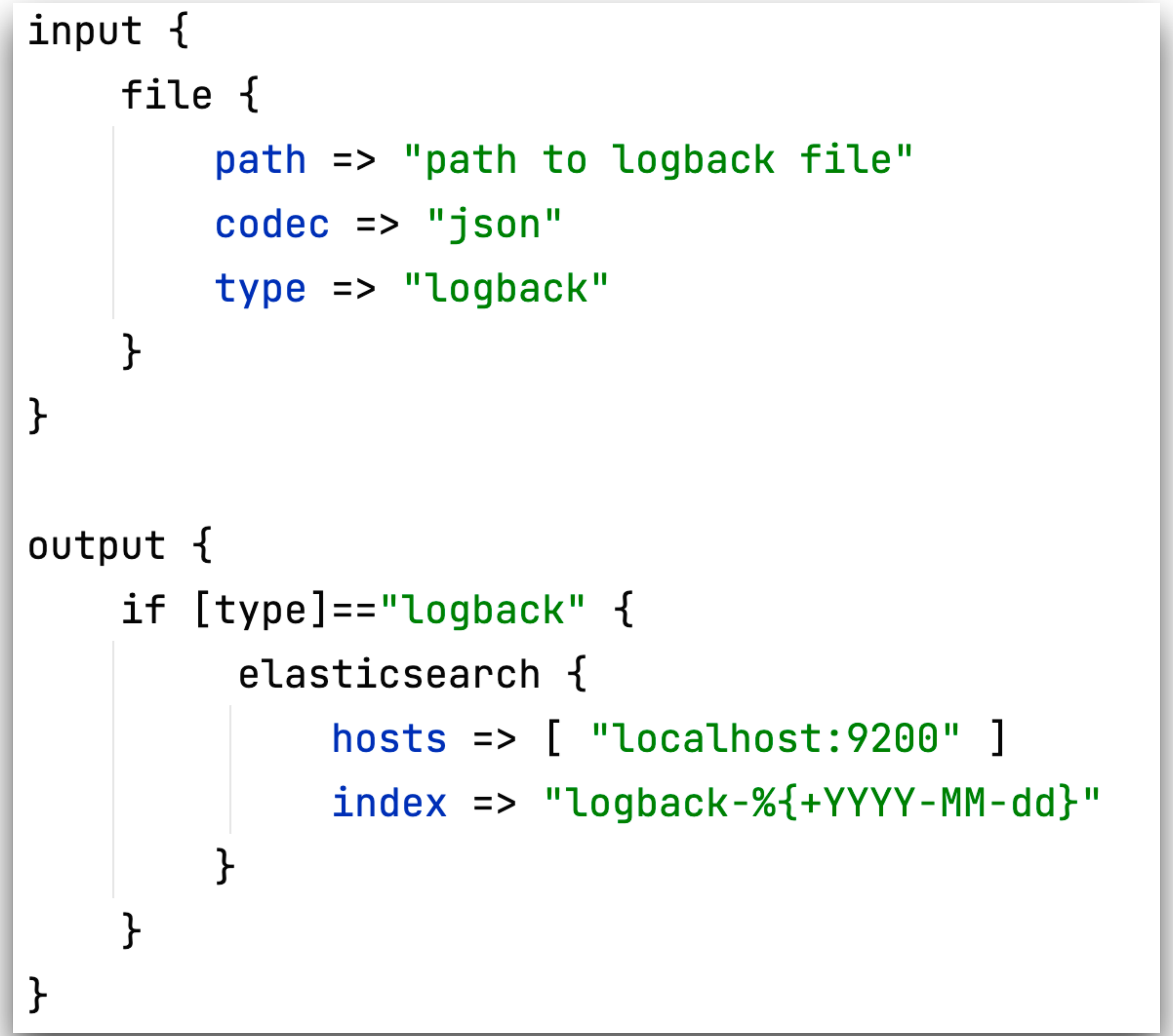


Working ELK stack



https://www.elastic.co/elastic-stack/

Config of logstash



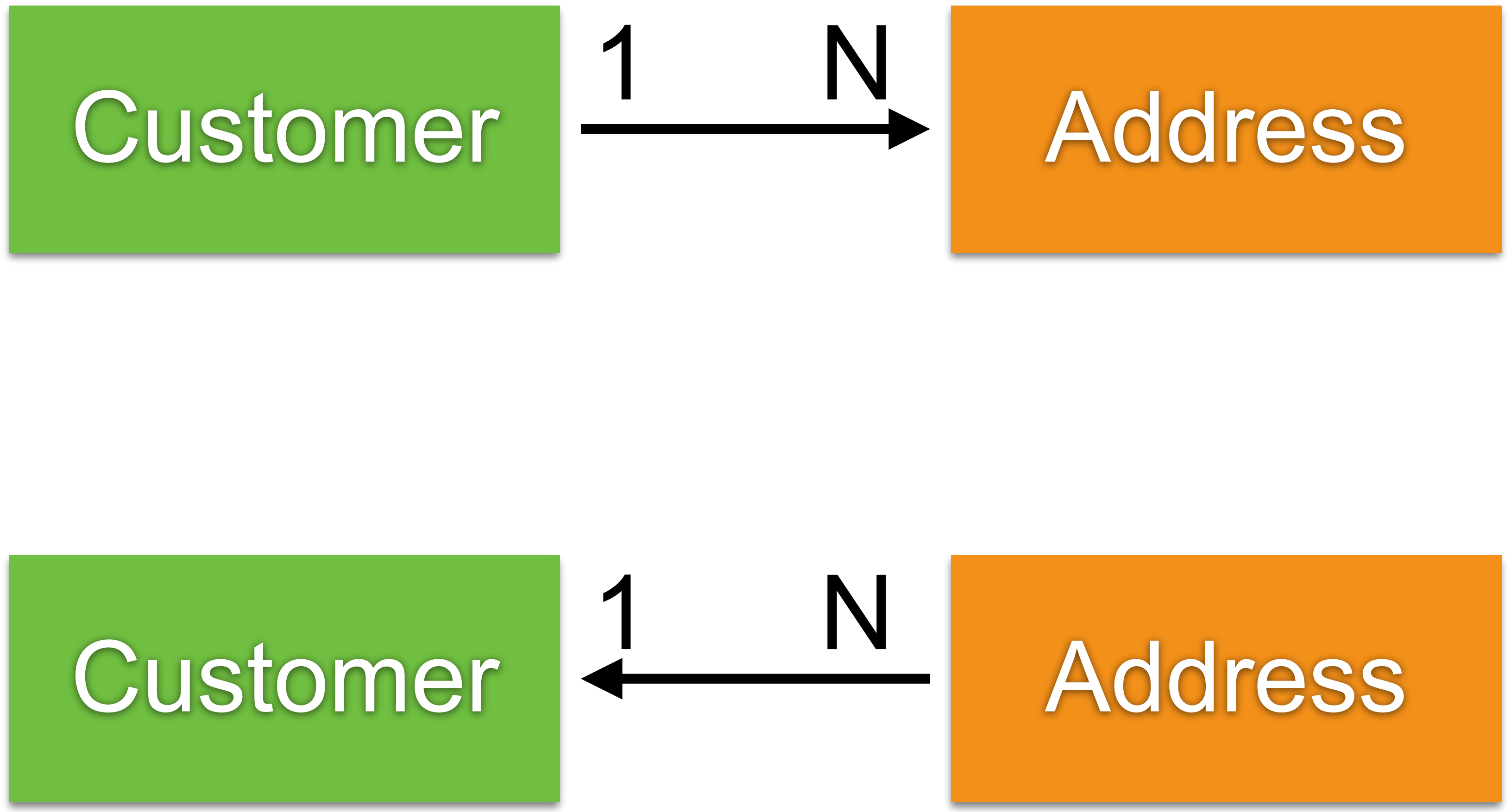
```
input {
    file {
        path => "path to logback file"
        codec => "json"
        type => "logback"
    }
}

output {
    if [type]=="logback" {
         elasticsearch {
             hosts => [ "localhost:9200" ]
             index => "logback-%{+YYYY-MM-dd}"
        }
    }
}
```

Entity’s relationship

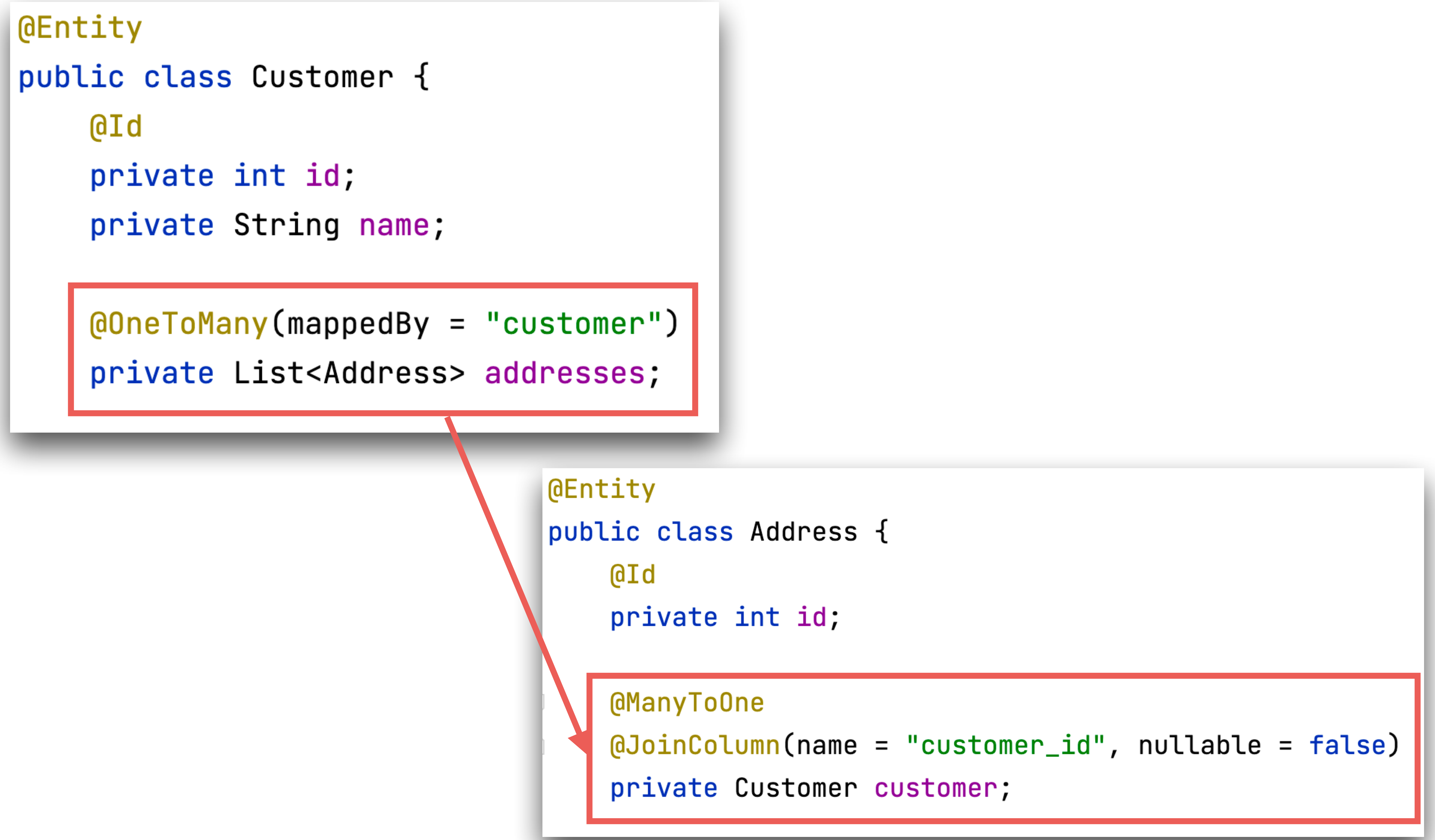
Relationship in JPA

Embedded (composite key)
OneToOne
OneToMany
ManyToOne
ManyToMay

One-to-many and Many-to-one



One-to-many and Many-to-one



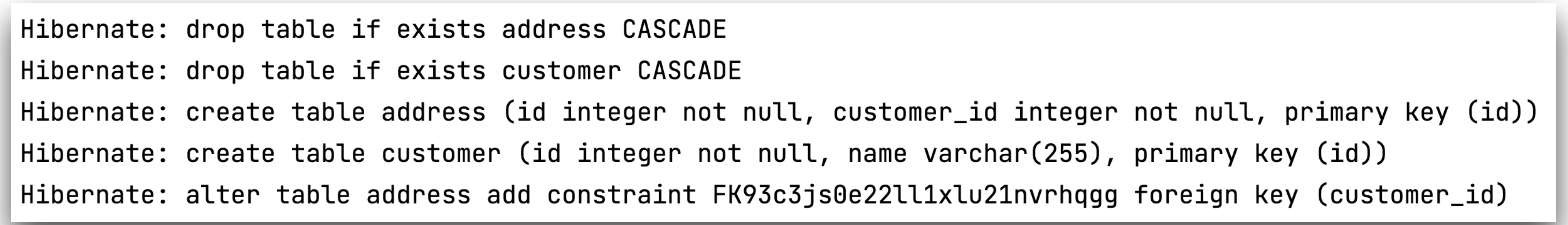
```java
@Entity
public class Customer {
    @Id
    private int id;
    private String name;

    @OneToMany(mappedBy = "customer")
    private List<Address> addresses;
```

```java
@Entity
public class Address {
    @Id
    private int id;

    @ManyToOne
    @JoinColumn(name = "customer_id", nullable = false)
    private Customer customer;
```

One-to-many and Many-to-one

application.properties

```
spring.jpa.show-sql=true
```



```
Hibernate: drop table if exists address CASCADE
Hibernate: drop table if exists customer CASCADE
Hibernate: create table address (id integer not null, customer_id integer not null, primary key (id))
Hibernate: create table customer (id integer not null, name varchar(255), primary key (id))
Hibernate: alter table address add constraint FK93c3js0e22ll1xlu21nvrhqgg foreign key (customer_id)
```

Load data from entity relation

FetchType.LAZY
FetchType.EAGER

FetchType is static, can’t change in runtime !!

Load data from entity relation

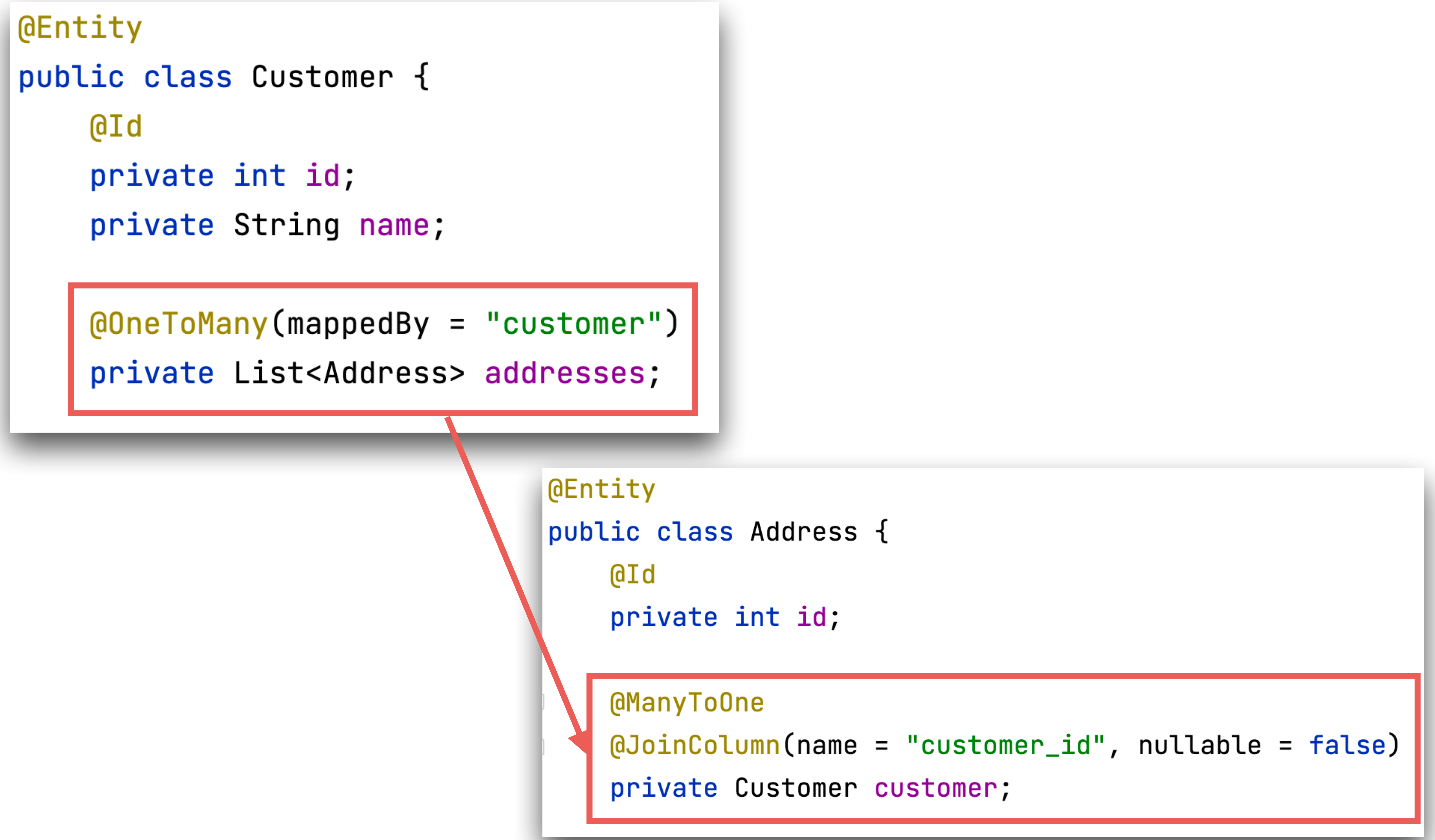
FetchType.LAZY

Default for @OneToMany, @ManyToMany

FetchType.EAGER

Default for @ManyToOne, @OneToOne

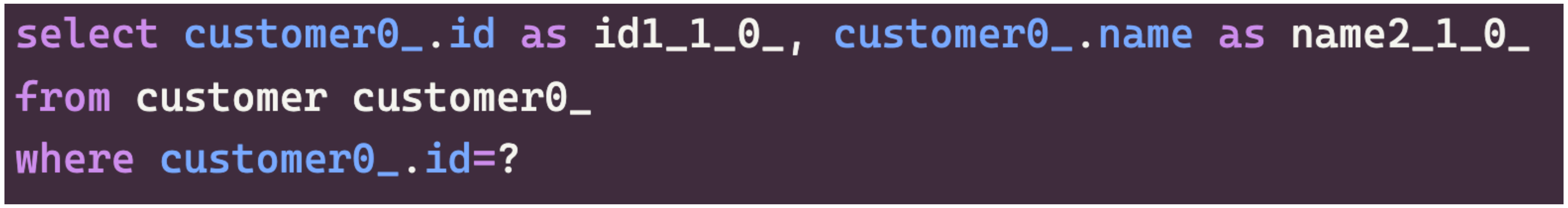
Load customer with address ?



```java
@Entity
public class Customer {
    @Id
    private int id;
    private String name;

    @OneToMany(mappedBy = "customer")
    private List<Address> addresses;
```

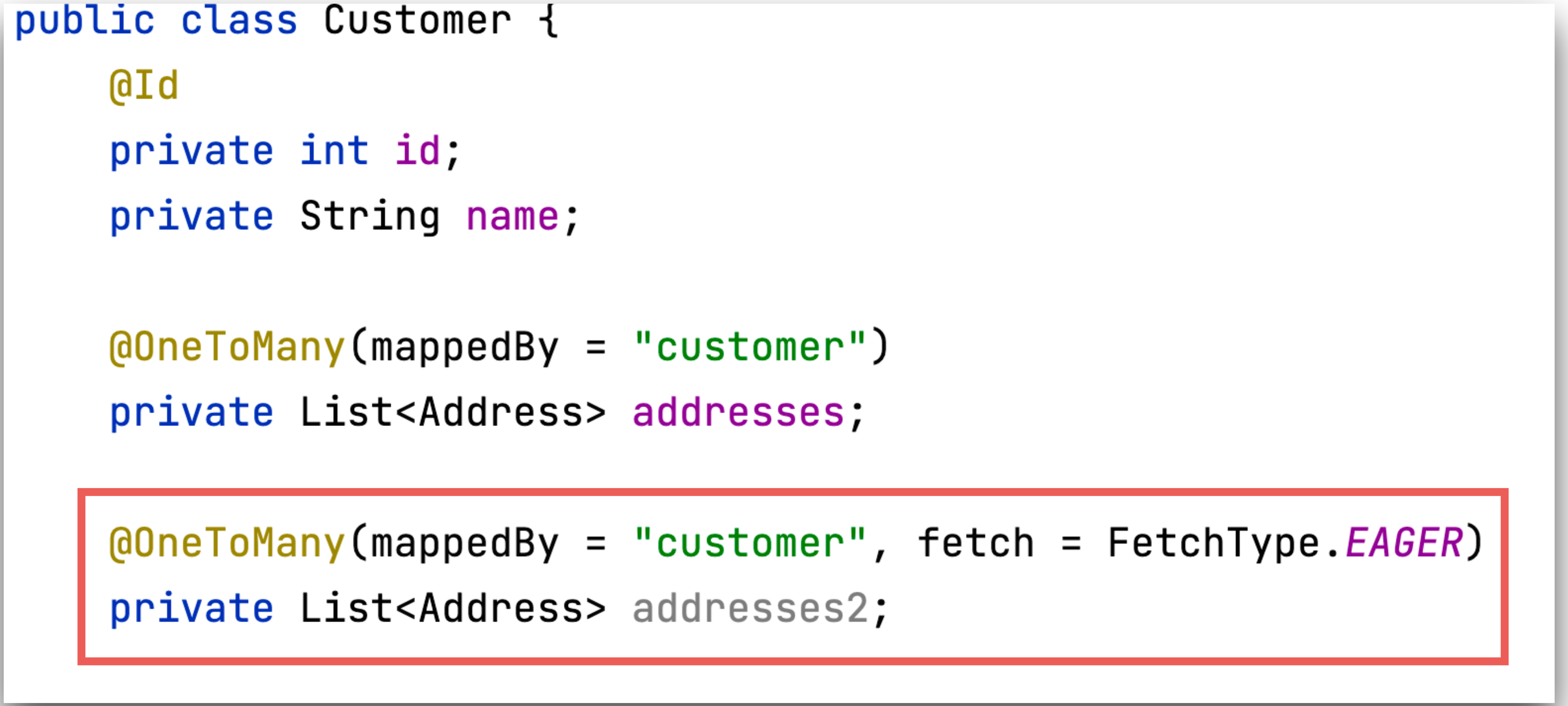
```java
@Entity
public class Address {
    @Id
    private int id;

    @ManyToOne
    @JoinColumn(name = "customer_id", nullable = false)
    private Customer customer;
```

Load customer with address ?



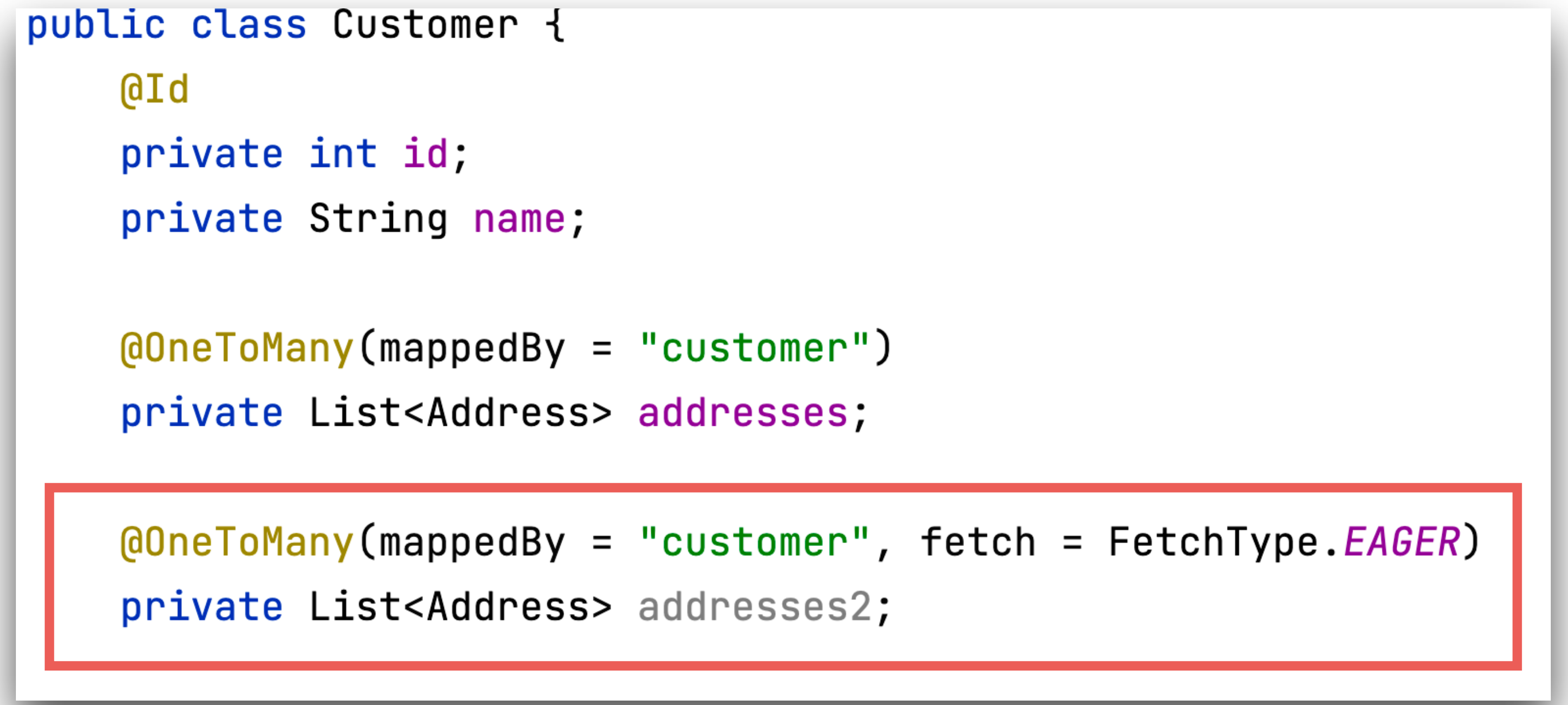
```
select customer0_.id as id1_1_0_, customer0_.name as name2_1_0_
from customer customer0_
where customer0_.id=?
```

What happen ?

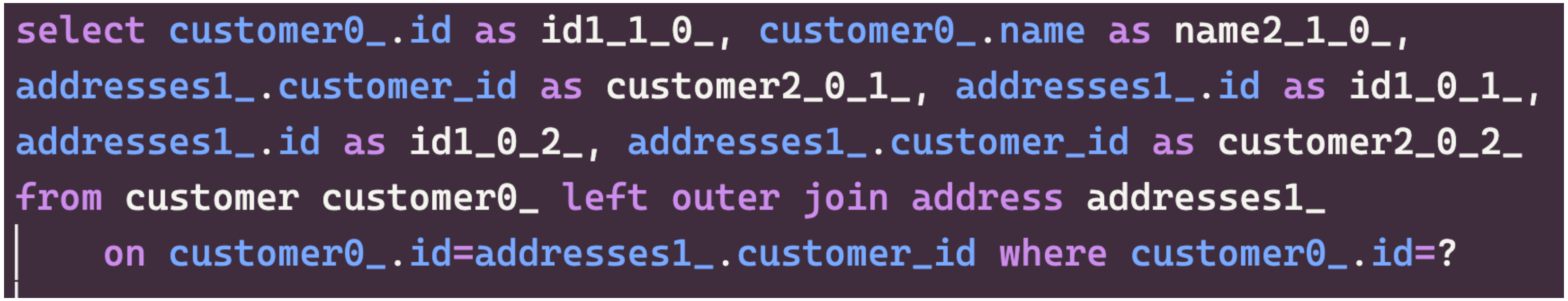


```
public class Customer {
    @Id
    private int id;
    private String name;

    @OneToMany(mappedBy = "customer")
    private List<Address> addresses;

    @OneToMany(mappedBy = "customer", fetch = FetchType.EAGER)
    private List<Address> addresses2;
```

What happen ?



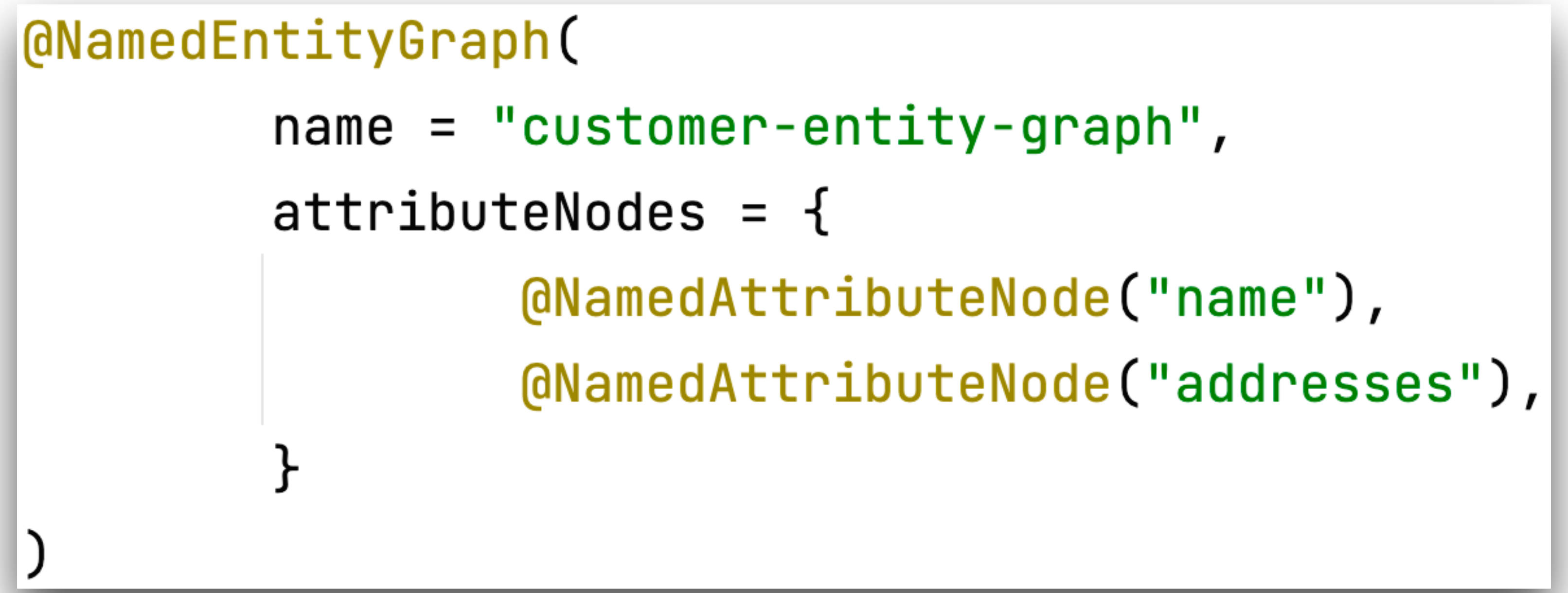
```
public class Customer {
    @Id
    private int id;
    private String name;

    @OneToMany(mappedBy = "customer")
    private List<Address> addresses;

    @OneToMany(mappedBy = "customer", fetch = FetchType.EAGER)
    private List<Address> addresses2;
```



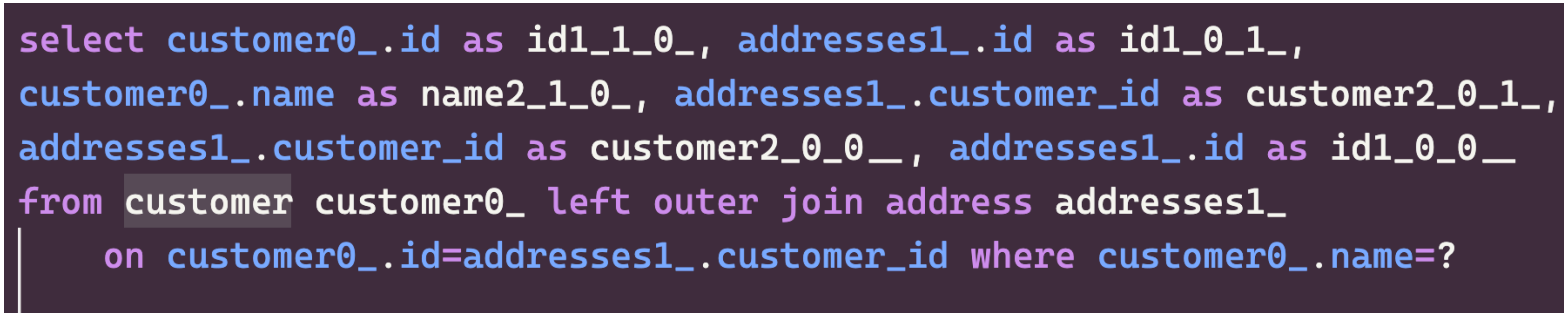
```
select customer0_.id as id1_1_0_, customer0_.name as name2_1_0_,
addresses1_.customer_id as customer2_0_1_, addresses1_.id as id1_0_1_,
addresses1_.id as id1_0_2_, addresses1_.customer_id as customer2_0_2_
from customer customer0_ left outer join address addresses1_
    on customer0_.id=addresses1_.customer_id where customer0_.id=?
```

FetchType is static, can’t change at runtime

Working with entity graph

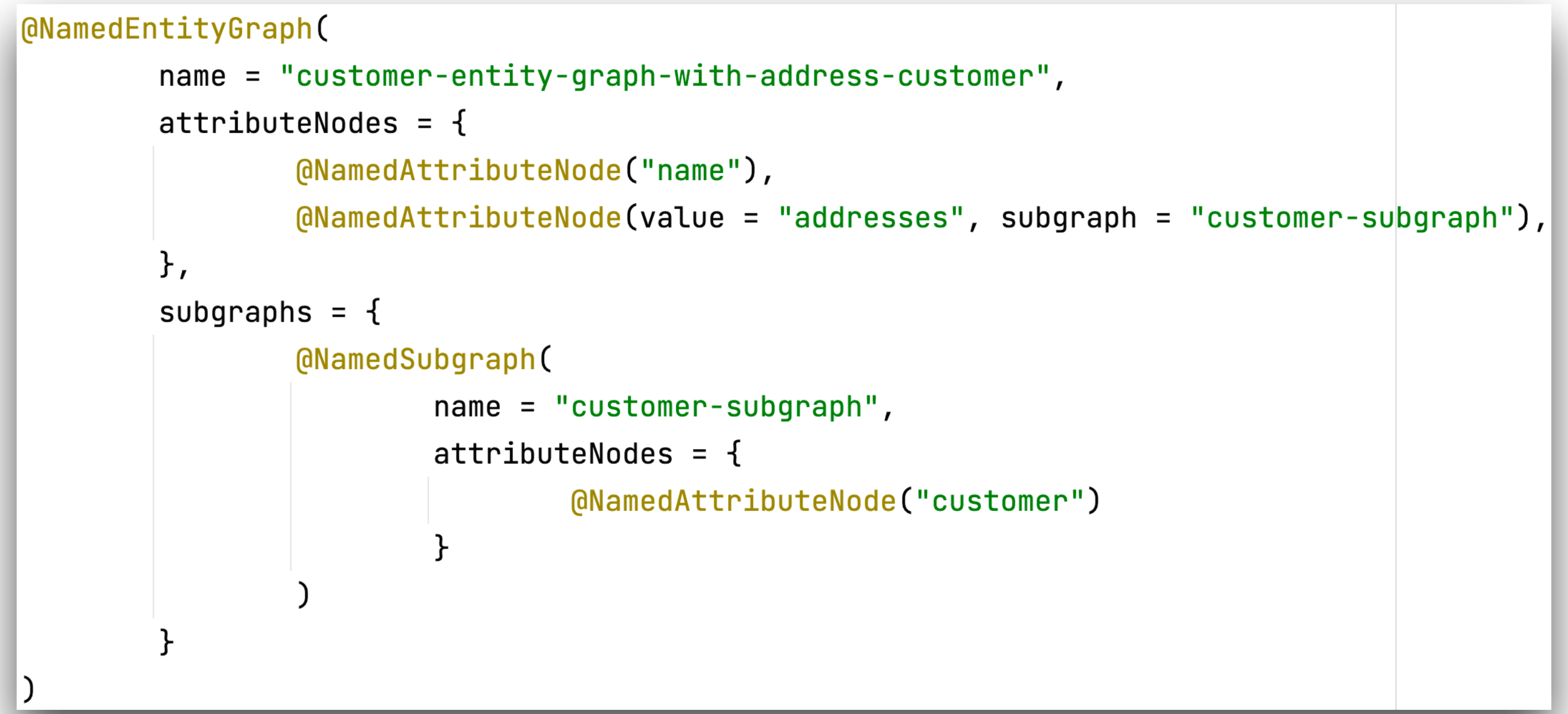


```
@NamedEntityGraph(
        name = "customer-entity-graph",
        attributeNodes = {
                @NamedAttributeNode("name"),
                @NamedAttributeNode("addresses"),
        }
)
```

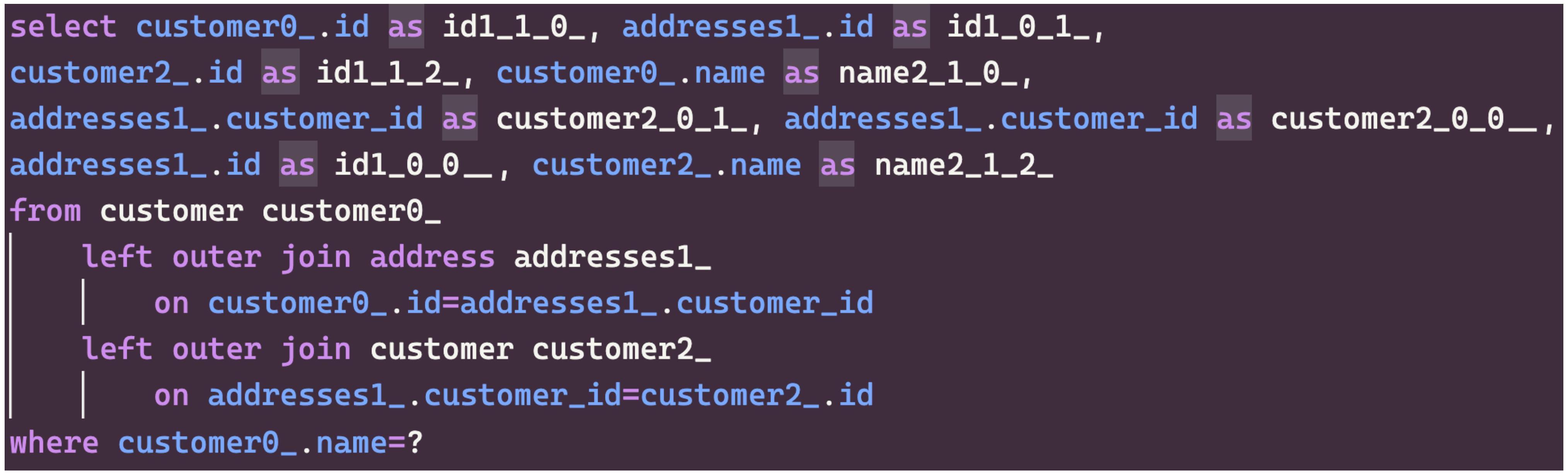


```
select customer0_.id as id1_1_0_, addresses1_.id as id1_0_1_,
customer0_.name as name2_1_0_, addresses1_.customer_id as customer2_0_1_,
addresses1_.customer_id as customer2_0_0__, addresses1_.id as id1_0_0__
from customer customer0_ left outer join address addresses1_
    on customer0_.id=addresses1_.customer_id where customer0_.name=?
```

Working with entity graph



```
@NamedEntityGraph(
        name = "customer-entity-graph-with-address-customer",
        attributeNodes = {
                @NamedAttributeNode("name"),
                @NamedAttributeNode(value = "addresses", subgraph = "customer-subgraph"),
        },
        subgraphs = {
                @NamedSubgraph(
                        name = "customer-subgraph",
                        attributeNodes = {
                                @NamedAttributeNode("customer")
                        }
                )
        }
)
```



```
select customer0_.id as id1_1_0_, addresses1_.id as id1_0_1_,
customer2_.id as id1_1_2_, customer0_.name as name2_1_0_,
addresses1_.customer_id as customer2_0_1_, addresses1_.customer_id as customer2_0_0__,
addresses1_.id as id1_0_0__, customer2_.name as name2_1_2_
from customer customer0_
    left outer join address addresses1_
        on customer0_.id=addresses1_.customer_id
    left outer join customer customer2_
        on addresses1_.customer_id=customer2_.id
where customer0_.name=?
```

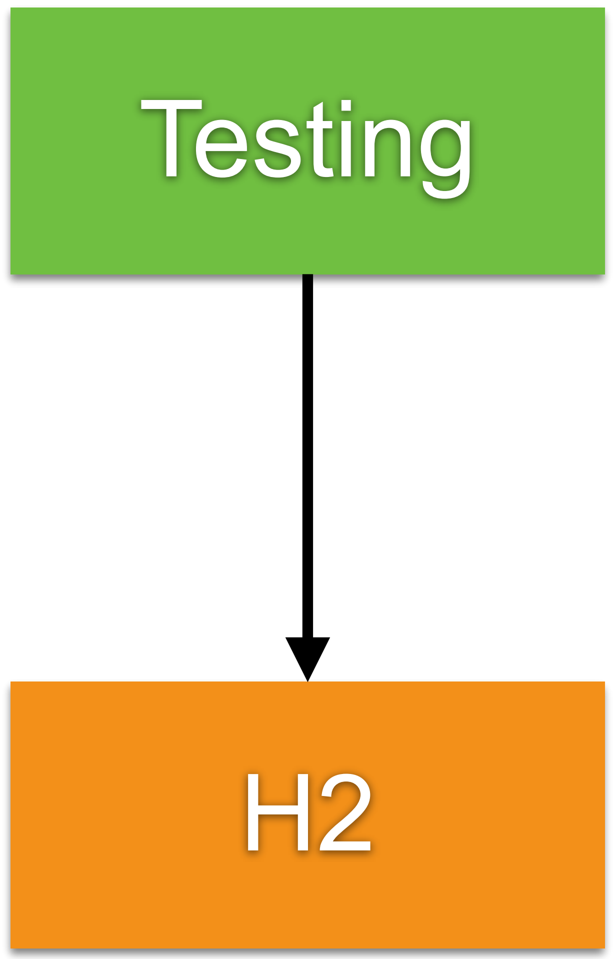
Working with entity graph

CustomerRepository.java

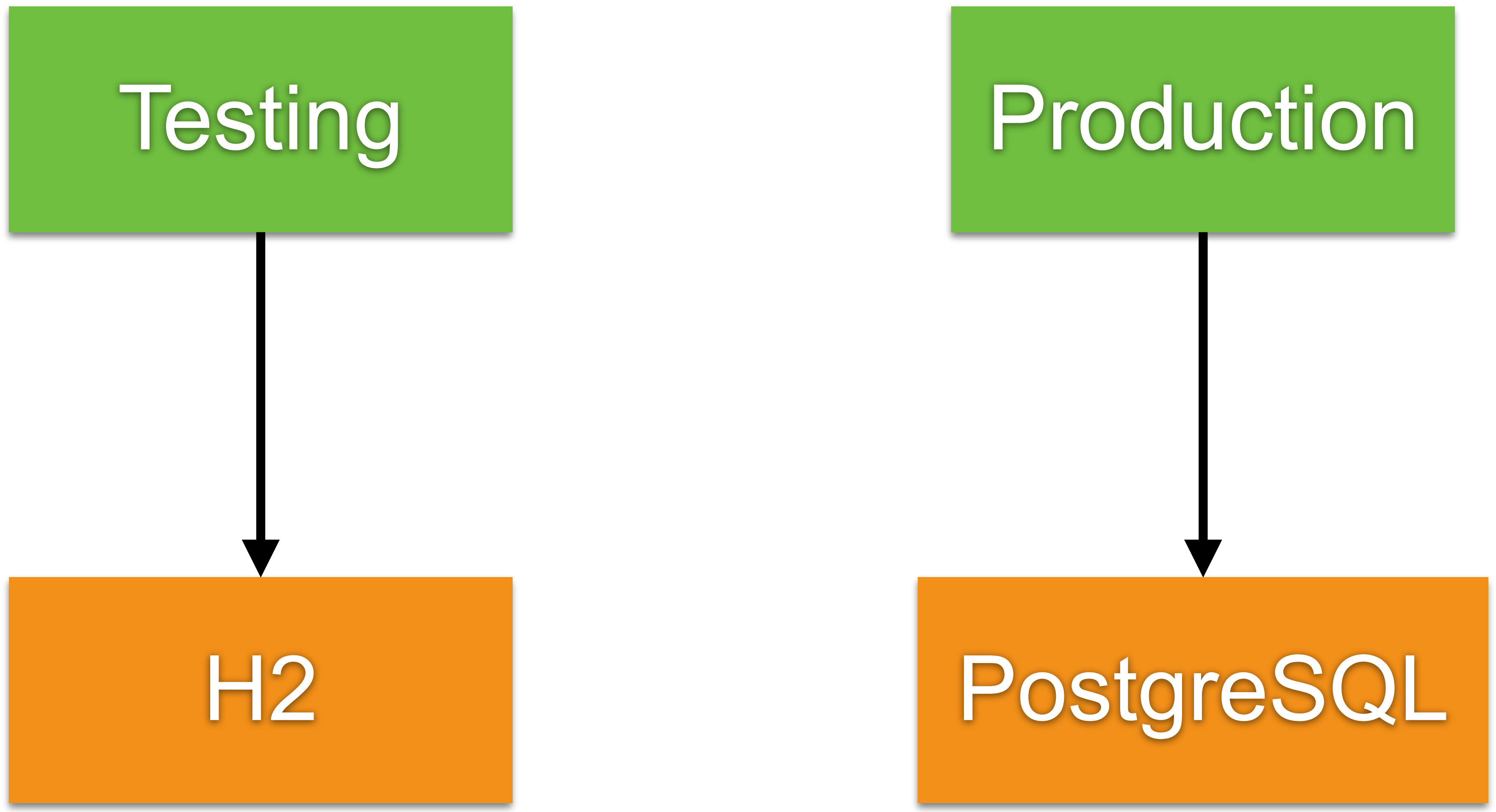
```java
import org.springframework.data.jpa.repository.EntityGraph;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomerRepository extends JpaRepository<Customer, Integer> {

    @EntityGraph(value = "customer-entity-graph-with-address-customer")
    Customer findByName(String name);

}
```

Working with database

Working with database



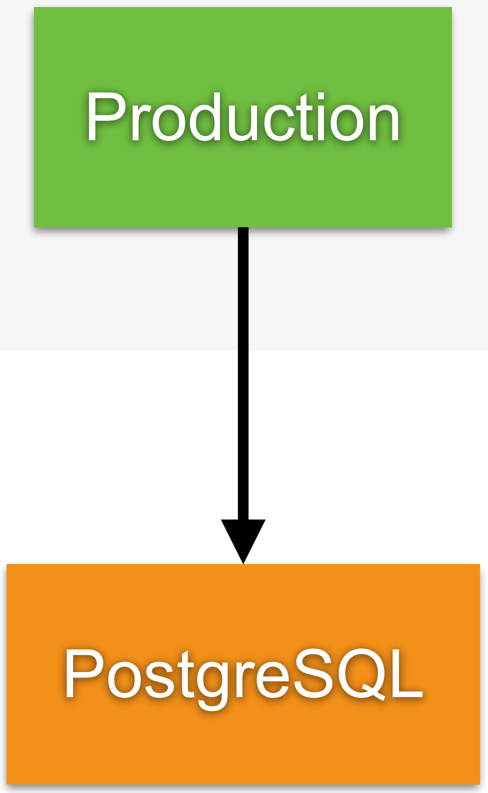
Working with database



Database Configuration

File src/main/resources/application.properties

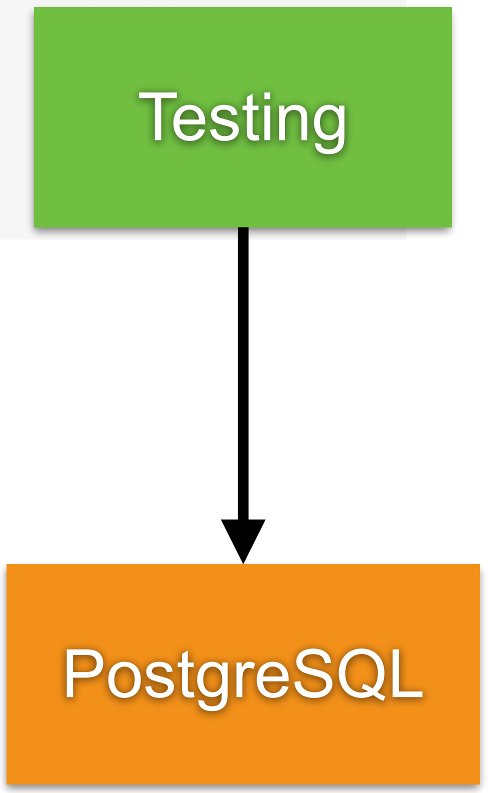
```
spring.datasource.url=${POSTGRES_URL:jdbc:postgresql://localhost:5432/demo}
spring.datasource.username=${POSTGRES_USER:postgres}
spring.datasource.password=${POSTGRES_PASS:password}

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update

spring.jpa.show-sql=true
```



Database Configuration

File src/test/resources/application.properties

```
## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=

# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto=update
```



Tuning database configuration

```
spring.datasource.hikari.minimumIdle=3
spring.datasource.hikari.maximumPoolSize=10
spring.datasource.hikari.poolName=SpringBootJPAHikariCP
spring.datasource.hikari.connectionTimeout=10000
spring.datasource.hikari.idleTimeout=100
spring.datasource.hikari.maxLifetime=120000
```

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#common-application-properties-data

Q/A